Amendments to the Specification:

Please replace the paragraph on page 14, lines 3-17 with the following new paragraph:

An operating system runs on processor **102** and is used to coordinate and provide control of various components within data processing system **100** in **Figure 1**. The operating system may be a commercially available operating system such as ~~Windows XP~~Windows XP™, which is available from Microsoft Corporation. An object oriented programming system such as [[Java]]Java™ may run in conjunction with the operating system and provides calls to the operating system from [[Java]]Java™ programs or applications executing on client **100**. "[[Java]]Java™" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented programming system, and applications or programs are located on storage devices, such as hard disk drive **126**, and may be loaded into main memory **104** for execution by processor **102.**

Please replace the paragraph on page 22, lines 12-29 with the following new paragraph:

Performance monitor unit **240** includes an implementation-dependent number (e.g., 2-8) of counters **241-242**, labeled PMC1 and PMC2, which are utilized to count occurrences of selected events. Performance monitor unit **240** further includes at least one monitor mode control register (MMCR). In this example, two control registers, MMCRs **243** and **244** are present that specify the function of counters **241-242**. Counters **241-242** and MMCRs **243-244** are preferably implemented as SPRs that are accessible for read or write via MFSPR (move from SPR) and MTSPR (move to SPR) instructions executable by CFXU **26**. However, in one alternative embodiment, counters **241-242** and MMCRs **243-244** may be implemented simply as addresses in I/O space. In another alternative embodiment, the control registers and counters may be accessed indirectly via an index register. This embodiment is implemented in the [[IA-64]]IA-64™ architecture in processors from Intel Corporation.

Please replace the paragraph extending from page 28, line 26 to page 29, line 5 with the following new paragraph:

Turning next to **Figure 6**, a diagram illustrating a bundle is depicted in accordance with a preferred embodiment of the present invention. Bundle **600** contains instruction slot **602**, instruction slot **604**, instruction slot **606** and template **608**. As illustrated, bundle **600** contains 128 bits. Each

instructions slot contains 41 bits, and template **608** contains 5 bits. Template **608** is used to identify stops within the current bundle and to map instructions within the slots to different types of execution units.

Please replace the paragraph extending from page 47, line 16 to page 48, line 28 with the following new paragraph:

The mechanism of the present invention provides for the construction and maintenance of call flow trees that may be accessed by an executing program for use in dynamic data area coverage. When a request is made for an allocation of data, such as a malloc, a routine is called to build trees. One methodology for determining the call stack is to walk the stack to determine the calling sequence at the time of the malloc. Another methodology it to use the hardware information generated through setting data access indicators. Techniques similar to that described in United States patent application entitled "Method and Apparatus for Determining Computer Program Flows Autonomically Using Hardware Assisted Thread Stack Tracking and Cataloged Symbolic Data", serial number [[_____]] 10/803,663, attorney docket no. AUS920030548US1 filed on [[_____]] March 18, 2004, which is incorporated herein by reference. This technique is used to identify the calling sequence; hereafter called call stack. This tree is maintained in memory and may be accessed through calls such as application programming interface (API) calls to a device driver which reads the call stack information for the current thread. The process for maintaining the hardware thread maintained call stack and to convert the addresses to symbolic names also is described in United States patent application entitled "Method and Apparatus for Determining Computer Program Flows Autonomically Using Hardware Assisted Thread Stack Tracking and Cataloged Symbolic Data", serial number [[_____]] 10/803,663, attorney docket no. AUS920030548US1 filed on [[_____]] March 18, 2004. The call stack retrieved from the device driver is sent to the arcflow program, which walks the stack into its thread oriented trees. This process is described in United States patent application, entitled "Method and System for Merging Event-Based Data and Sampled Data Into Postprocessed Trace Output", serial number 09/343,438, attorney docket no. AT9-98-850, filed on June 30, 1999. One approach involves indicating the start area, the end area, and one byte beyond the end area for flagging. Further, other statistics may also be kept in this memory area. Examples of other statistics involved using other hardware assist capabilities, the number of accesses, cache misses, cycles, etc. may be maintained.

Please replace the paragraph on page 56, lines 18-26 with the following new paragraph:

Referring back to step **2404**, if hardware decides not to prefetch data then the process returns to step **2408** as described above. In step [[**2302**]] **2402**, if metadata for prefetch is not associated with instruction, then the process proceeds to step **2408** as described above. In these examples, the hardware may send the instruction to a unit for processing before, at the same time, or after the hardware issues an indication that a prefetch should occur.

Please replace the paragraph on page 57, lines 5-11 with the following new paragraph:

Turning now to **Figure 25**, a flowchart of a process for illustrating metadata including an identification of what data is to be prefetched from the starting point is depicted in accordance with the preferred embodiment of the present invention. The process illustrated in **Figure 25** may be implemented into a ~~load store~~ load/store unit, such as ~~load store~~ load/store unit **228** in **Figure 2**.